Содержание

1 Лабораторная работа 4. Криптосистема openssl. Шифрование/подпись данных и создание PKI **1**
1.1 О криптосистеме OpenSSL 1
1.2 Поддерживаемые криптоалгоритмы 1
1.3 Симметричное шифрование-дешифрирование средствами openssl 2
1.3.1 Шифрование на основе вводимой пользователем парольной фразы 2
1.3.2 Дешифрирование зашифрованного файла 2
1.3.3 Рандомизация ключей шифрования 2
1.4 Создание закрытого ключа и запроса на подпись открытого ключа 3
1.4.1 Задание 1. Создание закрытого ключа и CSR-запроса для узла (FQDN-имя) 4
1.5 Создание самоподписанного сертификата 4
1.6 Создание собственного УЦ средствами openssl 5
1.6.1 Задание 2. Создание самоподписанного сертификата удостоверяющего центра 6
1.7 Подпись запроса на генерацию сертификата (CSR) 6
1.7.1 Задание 3. Подпись ключом CA CSR-запроса 6
1.8 Итоговое задание. Загрузка результата выполнения лабораторной работы 7

1. Лабораторная работа 4. Криптосистема openssl. Шифрование/подпись данных и создание PKI

1.1. О криптосистеме OpenSSL

- **OpenSSL** - это криптографическая библиотека и набор утилит командной строки, реализующих протоколы: **Secure Sockets Layer (SSL)** и **Transport Layer Security (TLS)**, а так же большого количества симметричных, асимметричных криптоалгоритмов и алгоритмов вычисления хэш-функций.
- Данная библиотека имеет утилиты командной строки, предназначенные для генерации закрытых ключей для различных асимметричных криптоалгоритмов и запросов на генерацию сертификатов - Certificate Signing Requests (CSR-запросов), выполнения процедур шифрования и цифровой подписи, управления сертификатами и выполнения прочих вспомогательных операций.
- Библиотека OpenSSL написана на C, однако существуют оболочки для иных языков программирования.

1.2. Поддерживаемые криптоалгоритмы

- Список поддерживаемых **симметричных** криптоалгоритмов

```
$ openssl list -cipher-algorithms
```

- Список поддерживаемых **симметричных** криптоалгоритмов

```
$ openssl list -public-key-algorithms
```

- Список поддерживаемых криптоалгоритмов вычисления **хэш-функций**

```
$ openssl list -digest-algorithms
```

1.3. Симметричное шифрование-дешифрирование средствами openssl

1.3.1. Шифрование на основе вводимой пользователем парольной фразы

- Шифрование файла <in-file> с использованием симметричного алгоритма , результат записывается в файл <out-file>

```
$ openssl <algo> -in <in-file> -out <out-file>
```

- Пример: шифрование файла /etc/hosts при помощи алгоритма 3DES и запись результата в текущий каталог

```
$ openssl des3 -in /etc/hosts -out hosts.des3
```

1.3.2. Дешифрирование зашифрованного файла

- Дешифрирование зашифрованных данных. Алгоритм дешифрирования должен соответствовать алгоритму шифрования. Вводимая с клавиатуры парольная фраза тоже

```
$ openssl des3 -d -in <in-file> -out <out-file>
```

- Пример: дешифрирование ранее зашифрованного файла hosts.des3 и запись в текущий каталог

```
$ openssl des3 -d -in hosts.des3 -out hosts
```

- *Примечание!* В указанном варианте шифрования используется очень слабая функция формирования ключа из вводимой с клавиатуры парольной фразы. Единственный способ повышения безопасности - использовать очень сложную парольную фразу. Так или иначе, для защиты конфиденциальных данных такой способ шифрования нежелателен

1.3.3. Рандомизация ключей шифрования

- *Примечание!* По-умолчанию генерация ключа из парольной фразы рандомизуется при помощи специальной добавки (salt)

- В этом можно убедиться, выполнив шифрования одного и того же открытого текста с использованием одного и того же алгоритма и парольной фразы.

Результаты двух шифрований можно вывести на экран при помощи утилиты **hexdump** и визуально сравнить.

1.4. Создание закрытого ключа и запроса на подпись открытого ключа

- **Запрос на подпись открытого ключа** или другими словами - запрос на генерацию сертификата (Certificate Signing Request - CSR) - форма хранения открытого ключа абонента с добавлением идентифицирующей абонента информации.

  Далее - CSR-запрос. Файл, в котором он сохранен - CSR-файл

  Является промежуточной формой хранения открытого ключа, после создания передается на удостоверяющий центр (CA), который преобразует CSR-запрос в сертификат, устанавливая свою электронную подпись на сочетании открытого ключа абонента и его идентифицирующей информации.

- **CN - Common Name** - должно быть указанно ваше доменное имя (для сертификата узла), для которого вы собираетесь использовать сертификат, также можно указать дополнительную информацию о вашей компании, адресе и организации, но это уже необязательно.

- Чтобы создать закрытый ключ и CSR-запрос следует выполнить команду **req** утилиты **openssl**

```
$ openssl req -newkey rsa:4096 -nodes -keyout file.key -out file.csr
```

- **-newkey** указывает, что нужно создать новую пару ключей, а в параметрах мы сообщаем асимметричный криптоалгоритм rsa и сложность 4096 байт,

- **-nodes** указывает, что шифровать закрытый ключ не нужно (для сертификатов узлов не шифруется, для пользовательских сертификатов - можно)

- **-keyout** - имя файла для создаваемого закрытого ключа

- **-out** - имя файла для создаваемого CSR-запроса

- Если уже есть закрытый ключ, то можно создать для него CSR-запрос

```
$ openssl req -key file.key -new -out file.csr
```

  Где **file.csr** - CSR-файл, **file.key** - файл с закрытым ключем

  -new указывает что нужно создать CSR-запрос.

- Во время создания вам нужно будет указать необходимую информацию для CSR-запроса, т.е. идентифицирующую информацию абонента.

- Кроме того, можно создать CSR-запрос из уже существующего сертификата и закрытого ключа, тогда не придется вводить информацию, т. к. она будет получена из сертификата.

```
$ openssl x509 -in file.crt -signkey file.key -x509toreq -out file.csr
```

Где **file.csr** - CSR-файл, **file.crt** - файл с сертификатом, **file.key** - файл с закрытым ключем

-x509toreq указывает, что нужно использовать сертификат для X509 для получения CSR. Если был получен сертификат от CA, то этот параметр не нужен.

- Для расшифровки CSR-запроса можно использовать команду

```
$ openssl req -text -in file.csr
```

Где **file.csr** - CSR-файл

1.4.1. Задание 1. Создание закрытого ключа и CSR-запроса для узла (FQDN-имя)

1. Выясните **имя вашего узла** с использованием команды **hostname**
2. Выполните генерацию **закрытого ключа** и **CSR-запроса**. Закрытый ключ запишите при этом в файл **<host>.key**, CSR-запрос - в файл **<host>.csr**, где <host> - имя узла, выясненное в п.1
3. В запросах утилиты генерации в обязательном порядке укажите в параметре **Common Name (CN)** полное доменное имя вашего узла. Для получения **полного доменного имени узла (FQDN)** дополните имя узла из п.1 доменным суффиксом, например spbstu.ru
4. В запросе адреса электронной почты, укажите **Ваш адрес почты**, под которой Вы зарегистрированы на портале дистанционного обучения dl.spbstu.ru
5. **Challenge password** в запросах оставьте пустым. Остальные запрашиваемые с клавиатуры параметры указывать необязательно.

- *Например:* Команда **hostname** выдает имя **deb10-1**, закрытый ключ сохраняем в файл **deb10-1.key**, CSR-запрос - в файл **deb10-1.csr**, в качестве **Common Name** указываем

```
deb10-1.spbstu.ru
```

6. Выполните расшифровку содержимого CSR-запроса, обратите внимание на значение поля **Subject(Абонент)**
7. Полученный файл CSR-запроса **<host>.csr** является одним из **результатов лабораторной работы**.

1.5. Создание самоподписанного сертификата

- Для создания самоподписанного сертификата выполняется подписание сертификата ключом, на основе которого он был создан

```
$ openssl x509 -signkey file.key -in file.csr -req -days 365 -out file.crt
```

- Можно объединить все в одну команду и сразу создать закрытый ключ, csr и подписанный сертификат

```
$ openssl req -newkey rsa:4096 -nodes -keyout file.key -x509 -days 365 -
out file.crt
```

- Можно создать самоподписанный сертификат openssl из существующего закрытого ключа без csr

```
$ openssl req -key file.key -new -x509 -days 365 -out file.crt
```

- **-new** запрос информации о csr у пользователя.
- Посмотреть расшифровку содержимого сертификатов можно следующим образом

```
$ openssl x509 -text -in file.crt
```

1.6. Создание собственного УЦ средствами openssl

- **Минимальная реализация** удостоверяющего центра (центра сертификации - CA) - это:

1. Защищенный парольной фразой **закрытый ключ**:
2. **Сертификат** с открытым ключом - должен быть подписан:
   - либо своим же **закрытым ключом** (т.е. самоподписанный сертификат) - в таком случае говорят об отдельном удостоверяющем центре
   - закрытым ключом **родительского удостоверяющего центра** - в данном случае образуется иерархия удостоверяющих центров
3. Инструменты для подписи поступающих запросов на генерацию сертификатов (например openssl)
4. Протоколы/процедуры - получения запросов от абонентов и передачи им сгенерированных сертификатов

- Выполнение создание ключей и одновременно подпись сформированного открытого ключа своим же закрытым, т.е. формирование самоподписанного сертификата. Тем самым выполняются приведенные выше требования **1** и **2**.

```
$ openssl req -newkey rsa:4096 -x509 -keyout ca.key -out ca.crt -days 3654
. . .
```

- **openssl req** - вызывается функция создания запроса сертификата
- **-newkey rsa:4096** - запрашивается генерация нового ключа RSA с длиной 4096 бит.
- **-x509** - создавать самоподписанный сертификат стандарта X509.
- **-keyout ca.key** - файл, куда сохранять закрытый ключ.
- **-out ca-.crt** - файл, куда сохранять подписанный сертификат.
- **-days 3654** - сертификат будет действителен примерно 10 лет, начиная от момента подписания.

- В диалоговом режиме утилита запрашивает:
  - **Enter PEM pass phrase**: парольная фраза для защиты закрытого ключа (методом шифрования)
  - Стандартные вопросы диалога создания CSR, в котором необходимо задать CN - имя создаваемого удостоверяющего центра
- В итоге выполнения команды созданы:
  - защищенный парольной фразой закрытый ключ удостоверяющего центра (ca.key)
  - самоподписанный сертификат удостоверяющего центра с его публичным ключом, идентификационной информацией удостоверяющего центра и сроком действия 10 лет (ca.crt)

1.6.1. Задание 2. Создание самоподписанного сертификата удостоверяющего центра

1. Выполните создание закрытого ключа и самоподписанного сертификата удостоверяющего центра. Самоподписанный сертификат сохраните в файл **ca.crt**
2. Укажите и запомните парольную фразу, которой будет зашифрован закрытый ключ удостоверяющего центра.
3. В качестве **CN (Common Name)** укажите **“Infsec Course CA”**,
4. В запросе адреса электронной почты, укажите адрес почты, под которой Вы зарегистрированы на портале дистанционного обучения
5. Посмотрите расшифровку содержимого сертификата и убедитесь, что записи **Subject(Абонент)** и **Issuer(Выдавший)** содержат одно и то же значение, т.е. сертификат самоподписанный
6. Полученный файл самоподписанного сертификата **ca.crt** является одним из результатов лабораторной работы.

1.7. Подпись запроса на генерацию сертификата (CSR)

- Для создания сертификата абонента необходимо выполнить установку цифровой подписи удостоверяющего центра на полученный от абонента CSR-запрос

```
$ openssl x509 -req -in file.csr -CA ca.crt -CAkey ca.key -
CAcreateserial -out file.crt
```

Где **file.csr** - CSR-запрос, **file.crt** - создаваемый сертификат, **ca.crt, ca.key** - сертификат и закрытый ключ удостоверяющего центра

1.7.1. Задание 3. Подпись ключом CA CSR-запроса

1. Выполните подпись CSR-запроса для **вашего узла**, полученного в **Задании 1** закрытым ключом удостоверяющего центра, созданного в **Задании 2**

2. Полученный в результате сертификат узла запишите в файл **<host>.crt**, где host - имя вашего узла, как было определено в **Задании 1**.
3. При запросе парольной фразы укажите ранее вводимую парольную фразу для выполнения дешифрирования приватного ключа.
4. Полученный файл с сертификатом узла является одним из результатов лабораторной работы.

1.8. Итоговое задание. Загрузка результата выполнения лабораторной работы

- В качестве результата выполнения лабораторной работы необходимо загрузить в Moodle следующие файлы:
  - файл CSR-запроса из Задания 1
  - файл самоподписанного сертификата удостоверяющего центра из Задания 2
  - файл с сертификатом узла из Задания 3